Channel-wise Subband Input for Better Voice and Accompaniment Separation on High Resolution Music

Haohe Liu, Lei Xie, Jian Wu, Geng Yang

Audio, Speech and Language Processing Group (ASLP@NPU), School of Computer Science, Northwestern Polytechnical University, Xi'an, China

haoheliu@gmail.com, {lxie,jianwu,gengyang}@nwpu-aslp.org

Abstract

This paper presents a new input format, channel-wise subband input (CWS), for convolutional neural network (CNN) based music source separation (MSS) models in the frequency domain. We aim to address the major issues in CNN-based high-resolution MSS model: high computational cost and weight sharing between distinctly different bands. Specifically in this paper, we decompose the input mixture spectra into several bands and concatenate them channel-wise as the model input. The proposed approach enables effective weight sharing in each subband and introduces more flexibility between channels. For comparison purposes, we perform voice and accompaniment separation (VAS) on models with different scales, architectures, and CWS settings. The result shows that the CWS input is beneficial in many aspects. Among all our experiments, it enables models to obtain a 6.9% performance gain on average. With even a smaller number of parameters, less training data, and shorter training time, our MDenseNet with 8-bands CWS input still surpasses the original MMDenseNet with a large margin. Moreover, CWS also reduces computational cost and training time to a large extent.

Index Terms: voice and accompaniment separation, deep learning, subband, music source separation

1. Introduction

Music Source Separation (MSS) has raised much interest in recent years. The goal of the task is blindly separate sources from a mixed track, for example vocal, drums, bass and accompaniment. In this paper, we particularly focus on the voice and accompaniment separation (VAS) from a mixture. As a practical tool, separating these two components allows us to remix, suppress or up-mix the sources [1]. VAS can also facilitate automatic transcription, karaoke tracks generating as well as music information retrieval [2].

High-resolution music usually sounds better but suffers from high computational cost in the VAS task. For example, 44.1kHz is a commonly used sample rate for music, while many high-quality format may be up to 48kHz or even higher. However, due to the high-computational cost, many of the current VAS studies perform downsampling in advance. For instance, the approach using M-U-Net [3] downsamples the audio to 10.88kHz before processing and Dense-Unet only works on 16kHz music in [4]. The downsampling process seriously affects the auditory quality to the separated vocal and accompaniment in practical applications.

Convolutional Neural Network (CNN) has shown tremendous success in multiple fields, especially image-related tasks. The input data for these tasks, such as image classification, usually suffer from problems that the position of a certain object is not fixed. A quick solution from CNN are mechanisms like local receptive fields and shared weights [5]. These two features enable CNN to become position invariant, which means once a feature has been detected, its exact location becomes less important [5]. In audio processing, most of the state-of-the-art (SOTA) MSS models are also based on convolutional networks, like Deep-Unet [6], which have shown considerable improvements over the traditional methods.

Although CNN-based architecture has demonstrated effectiveness on MSS tasks in frequency domain, it still has apparent limitations. Frequency spectrogram based SOTA models trained on high-resolution audio, e.g. TFC-TIF [7], usually take the whole magnitude spectrogram as the input feature. In this case, they assume each frequency band has filter parameters to share and are equally important. However, local patterns are usually different between bands [8], as can be seen in Fig 1. This means different bands do not necessarily need the same set of filters (in CNN) for parameter sharing. So, treating different frequency bands differently might better facilitate the separation process.

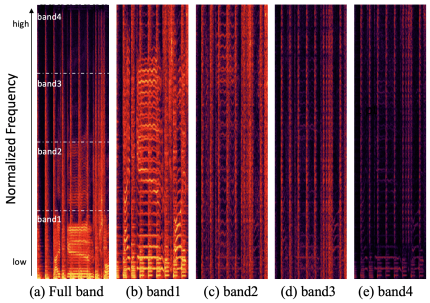

Figure 1: *Comparison between different bands. Lower frequency band contains more energy, long sustained sound, fundamental frequency and harmonic series. While higher band, mostly percussive signals and low-energy resonance, contains less energy and less complex information.*

Some prior efforts have already emphasized the difference between bands. In [9], Taghia *et al.* first took the subband decomposition, and then they used a hybrid system of empirical mode decomposition [10] and principle component analysis to construct artificial observations from the single mixture. Finally a synthesis process was used to reconstruct full band signal. Takahashi *et al.* also noticed the problem of global kernel sharing. They pointed out that the global weight sharing works well on natural photos, in which local pattern appears in any position of the input [8]. But this is not the case for audio. To handle that problem, they design the dedicated MDenseNets for each frequency band and a full band MDenseNet for full band

rough structure [8]. This model outperformed the state-of-the-art result on SiSEC 2016 competition by a large margin [11].

In this paper, we propose a new input format for the MSS model in the frequency domain: channel-wise subband (CWS) input. Different from the band-dedicated approach in [8, 12], our method can handle both sub-bands and full-band in a single model, which makes CWS-based model highly efficient, less complex, and easier to use. We extensively evaluate our method on MDenseNet, UNet [6] with different scales, and three kinds of subband settings on musdb18hq [13]. We also test our approach on a larger internal dataset. Results show that the model with CWS input not only outperforms the model without CWS by a large margin, but also boosts the speed of model training and separation.

2. Methodology

Given the raw music signal $y(n)$, our goal is to separate a set of source signals $x(n) = \{x_j(n)\}, j \in \{1, \cdots, J\}$. In this paper, we focus on VAS task. Thus $J = 2$ and $x_1(n), x_2(n)$ are vocal and accompaniment, respectively. In the time domain, the observed mixture signal is modeled as:

$$y(n) = x_1(n) + x_2(n), \tag{1}$$

and in the frequency domain, it equals to

$$Y(t, f) = X_1(t, f) + X_2(t, f), \tag{2}$$

where t, f index time and frequency axis. Y, X_1, X_2 are short-time fourier transform (STFT) of the mixture signal $y(n)$ and the source signal $x_1(n), x_2(n)$. In this section, we will brively introduce the UNet and MMDenseNet and the propose the analysis-synthesis scheme and CWS feature format.

2.1. UNet

Fig 2 depicts the structure of UNet [14] we used in this paper with a scale of $s = 5$. It takes the mixture spectra $Y(t, f)$ as input and outputs the Time-Frequency Mask [15] $M_j(t, f)$ for source j, which has identical size with the input. The *in-conv* block firstly expands the input channel to 64. After that, we go through a series of convolution blocks, down/upsampling layers with skip connections. Each convolution block consists of two series of 2D convolution layer, batch normalization and rectified linear units [16]. We use 3×3 kernel size in convolution layers, with the padding value of 1 to make sure that the frequency and time dimension will not be changed by the convolution operations. We use max-pooling and linear interpolation to scale down and up the feature map by a factor of 2. Skip connections were added between down and up path. The input feature of each convolution layer in the up path is concatenated with the same scale output in the down path. To constraint the mask value between $[0, 1]$, the sigmoid function is added on the

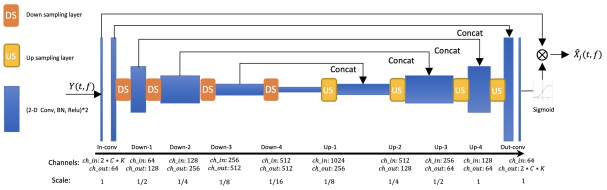

Figure 2: *Unet Structure used in this paper*

model's output. Finally the source spectrogram is obtained by multiply the mask and the mixture spectrogram:

$$\hat{X}_j(t, f) = M_j(t, f) \cdot Y_j(t, f) \tag{3}$$

The final separated music signal $\hat{x}_j(n)$ is obtained through inverse short-time fourier transform (iSTFT) using the source spectrogram estimation in Eq. (3).

Blocks	MMDenseNet			MDenseNet
	low(k,L)	high(k,L)	full(k,L)	
conv((t,f),ch) dense1	conv((3,4),32) (14,4)	conv((3,3),32) (10,3)	conv((3,3),32) (6,2)	conv((3,3),32) (14,4)
dense2	conv1d,pool (16,4)	conv1d,pool (10,3)	conv1d,pool (6,2)	conv1d,pool (16,4)
dense3	conv1d,pool (16,4)	conv1d,pool (10,3)	conv1d,pool (6,2)	conv1d,pool (16,4)
dense4	conv1d,pool (16,4)	conv1d,pool (10,3)	conv1d,pool (6,2)	conv1d,pool (16,4)
dense5	t.conv (16,4)	t.conv (10,3)	t.conv (6,2)	conv1d,pool (16,4)
dense6	t.conv (16,4)	t.conv (10,3)	t.conv (6,2)	t.conv (16,4)
dense7	t.conv (16,4)	t.conv (16,3)	t.conv (6,2)	t.conv (16,4)
concat/dense8	freqency axis		–	t.conv (16,4)
concat/dense9	channel axis			t.conv (16,4)
out	DenseBlock(k=32,L=2)			
	Conv2d(ch_in=32,ch_out=2*C*K)			

Table 1: *Details of MMDenseNet and MDenseNet.*

2.2. MMDenseNet

With only about 0.3M parameters and state-of-the-art performance, MMDenseNet [8] is currently one of the most effective models for audio source separation. It utilizes the characteristics in each frequency band to design different MDenseNets dedicated for each band. The frequency axis size of band-dedicated MDenseNet will be reduced to f/K for it only process data within that specific band. The output of these MDenseNet, $\hat{X}_k(t, f/K)$, are concatenated in the frequency axis to recover full band prediction $\hat{X}_{\text{sub}}(t, f)$. To capture the global rough structure, MMDenseNet also have a MDenseNet for the full band. Then the output of the full band MDenseNet $\hat{X}_{\text{full}}(t, f)$, is concatenated with $\hat{X}_{\text{sub}}(t, f)$ and pass through a final denseblock to recover the final prediction $\hat{X}(t, f)$. Each MDenseNet inside MMDenseNet can be designed independently according to its function and data complexity. MMDenseNet is highly parameter efficient due to the use of denseblocks [17] and skip-connection between denseblocks. Inside denseblock, the input of Denselayers is the concatenation of previous layers' output or the skip connection of the former dense block. This scheme enables model to reuse feature effectively, thus it is highly parameter efficient. The MMDenseNet we use in this paper has a scale of 4, and have three MDenseNets, as is shown in Table 1. The detail of MMDenseNet was described in [8].

In this paper, we also conduct experiments on MDenseNet, both with CWS and without CWS. The structure of MDenseNet we use are shown in Table 1. For a fair comparison with MMDenseNet, we add two additional two dense blocks to the low part of MMDenseNet to form the MDenseNet we use. In this way, the scale of our MDenseNet is five and the total parameter number is 0.27 M.

2.3. Channel-wise Subband Input

We follow the method in [18] for subband decomposing and signal reconstruction in analysis and synthesis procedure. Both

analysis and synthesis include a group of finite impulse response (FIR) uniform filter banks. We design three sets of analysis filter banks $H_k(e^{j\omega})$ and coresponding synthesis filters $G_k(e^{j\omega})$, where $k \in 1, ..., K$ stands for the number of subbands. The design of these filters follows the procedure in [19]. We use $y_k(n)$ to denote the output of $H_k(e^{j\omega})$. After downsampling, the sample rate of $y_k(n)$ is $\frac{1}{K}$ of $y(n)$.

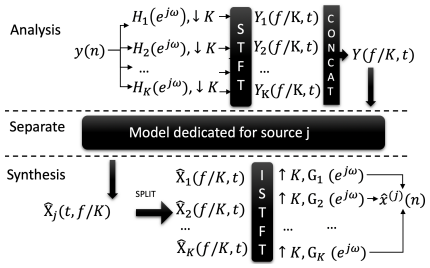

Figure 3: *Channel-wise subband input*

Though the total volume of input feature does not changed, the channel-wise concatenation of subbands is a better input format for the frequency domain model. Here we give a simple explaination. Using $\beta_\mu^{(l)}$ to denote the output feature map in l-th layer, μ channel, and $S_{\lambda,\mu}^{(l-1)}$ to stand for the λ-th convolution filters in layer $l-1$, which output is the μ-th channel in l-th layer. The 2D convolution layer can be discribed as

$$\begin{aligned} o_\mu^{(l)}(i,j) &= \sum_\lambda \beta_\lambda^{(l-1)} * S_{\lambda,\mu}^{(l-1)} \\ &= \sum_\lambda \sum_m \sum_n \beta_\lambda^{(l-1)}(i-m, j-n) S_{\lambda,\mu}^{(l-1)}(m,n) \end{aligned} \tag{4}$$

$$\beta_\mu^{(l)}(i,j) = \phi\left(o_\mu^{(l)}(i,j)\right) \tag{5}$$

From Eq. (4), we can observe that internal variable $o_\mu^{(l)}$ is the linear product and sum of $\beta_\lambda^{(l-1)}$ and $S_{\lambda,\mu}^{(l-1)}$. Thus we can not only view the convolutional kernel as feature extractor, but also weight between different channels. For example, if some filters in $S_\mu^{(l-1)}$ are set to all zero, then the corresponding channels in $\beta_\mu^{(l-1)}$ will not able to pass their value to $\beta_\mu^{(l)}$. In this way, the μ-th channel in feature map $\beta^{(l)}$ can select the exact channels to be used in the previous layer. In this way, the channel-wise concatenation enable model to assign different capability on channel dimension, which is helpful to make the model highly efficient.

After the analysis process, we performe STFT for each $y_k(n)$ and the result is denoted as $Y_k(f/K,t)$. Here the sample rate of $y_k(n)$ reduced by a factor of K, $K \in \{2,4,8\}$. So the size of frequency axis will also be reduced K fold, which will be detailed in Section 2.4. Then we concatenate $Y_k(f/K,t)$ along the subband dimension

$$Y = \left(\left[Y_1^1, Y_2^1, ..., Y_K^1, Y_1^2, Y_2^2, ..., Y_K^C\right]\right) \tag{6}$$

to form the input feature of the network. Since the data we used in this paper are all stereo, C equals to 2 here. The subscript k indexes subband and we omit the $(f/K,t)$ part in this equation for simplicity. Also we should clarify that Y is not a magnitude spectrogram as used in conventional frequency domain models [20, 6, 4, 21], but the complex-valued STFT matrices. This method is explored by [7] with considerable improvements. We treat different bands as different channels so that model can both learn different channels independently and incorporate bands' features in a deeper layer.

The synthesis procedure is a reverse version of the analysis. We split the network output $X_j(f/K,t)$ channel-wise as the prediction of each subband. After iSTFT, we pass the result through a set of synthesis filters to reconstruct source signal j.

2.4. Model Training

The synthesis procedure is not performed during training and the loss function is defined as the sum of two components

$$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_c, \tag{7}$$

where $\mathcal{L}_1$ is the L_1 norm and $\mathcal{L}_c$ denotes conservation loss. Conservation loss could help when two dedicated models were trained jointly because it follows the basic model in Eq. (2) and unites two independent dedicated-models. Each loss function measures the mean absolute error between network output $\hat{X}_j(f/K,t)$ and the coresponding reference magnitude:

$$\begin{aligned} \mathcal{L}_1 &= \sum_{j=1}^{2} \sum_{t,f} \left| \hat{X}_j(f/K,t) - X_j(f/K,t) \right| \\ \mathcal{L}_c &= \sum_{j=1}^{2} \sum_{t,f} \left| \hat{X}_j(f/K,t) - Y_j(f/K,t) \right| \end{aligned} \tag{8}$$

We perform validation with every two hours of the training data and stop the training progress if no validation improvement exists in 20 consecutive epochs. All the models are trained using Adam optimizer [22] with a initial learning rate of 0.001 and a dropout rate of 0.1. The learning rate decays every thirty hours of training data with a decay rate of 0.87. The STFT matrices with a FFT size of 32 ms and a hop size of 8 ms are used as the model input. The actual frame length and shift size (in number of the samples) are automaticly calculated with the sample rate of the input audio.

3. Experiments

In this section, we will describe the dataset and evaluation metrics used in this paper. The experimental comparison and analysis of the advantage of CWS will also be discussed.

3.1. Dataset

We mainly conduct experiments on the publicly available musdb18hq dataset [13]. It has a training set with 100 songs and a test set of 50 songs. We choose 14 songs from training set as the validation set, the same as the definition in python package *musdb*[1]. To explore the limitation of the data, we also trained our model on an internal training set named *aslp*. It has additional 617 songs of pure vocal and 1496 songs of pure instrument, which are collected from the internet. Although some of them may not be clean, experiments show that using additional data improves the separation performance. We followed the steps in [21] for data augmentation. During the training stage, we randomly select, chunk, and mix vocal and instruments and multiply two streams with a scaling factor randomly sampled between 0.6 and 1.0. All the songs in *musdb18hq* or *aslp* are stereo and the sample rate is 44.1 kHz.

[1] https://github.com/sigsep/sigsep-mus-db

Table 2: *Model comparison in terms of various metrics on musdb18hq test set*

	GFLOPs	Params (M)	Train (h)	SAR (A)	SAR (V)	SDR(A)	SDR (V)	SIR (A)	SIR (V)	Average
UNET-5	182.81	13.3	61	14.20	4.32	14.62	3.16	20.89	12.61	11.63
+CWS$_{K=2}$	91.90	13.3	36	14.10	**4.97**	**15.19**	**4.23**	**21.98**	11.99	**12.08**
+CWS$_{K=4}$	46.44	13.3	26	**14.23**	**5.05**	**15.56**	**4.35**	**22.54**	12.07	**12.30**
+CWS$_{K=8}$	23.71	13.3	15	14.04	**4.73**	**15.72**	**4.31**	**22.00**	11.58	**12.06**
MMDN	27.63	0.33	59	13.22	3.73	14.50	3.12	21.18	11.73	11.25
MDN	37.42	0.27	32	13.94	3.35	13.90	2.59	19.40	10.56	10.62
+CWS$_{K=2}$	19.03	0.27	27	**13.96**	**4.11**	**15.60**	**3.65**	**21.30**	**11.35**	**11.66**
+CWS$_{K=4}$	9.67	0.27	26	**14.10**	**4.00**	**15.28**	**3.86**	**20.91**	**12.03**	**11.70**
+CWS$_{K=8}$	5.07	0.27	10	**13.98**	**3.85**	**15.67**	**4.17**	**20.68**	**11.75**	**11.68**
UNET-6	220.73	53	73	13.34	4.45	14.42	3.28	23.14	9.52	11.36
+CWS$_{K=2}$	110.86	53	33	**14.15**	**4.73**	**14.62**	**3.92**	22.43	**11.50**	**11.89**
+CWS$_{K=4}$	55.92	53	23	**14.39**	**5.22**	**16.02**	**4.79**	22.63	**12.10**	**12.53**
+CWS$_{K=8}$	28.46	53	19	**14.01**	**4.86**	**15.97**	**4.95**	22.63	**11.46**	**12.31**
BD-UNET-6	220.73	53	149	13.87	4.79	15.20	3.94	22.73	11.33	11.98
+CWS$_{K=2}$	110.86	53	92	**14.24**	**4.85**	**15.44**	**4.34**	**22.79**	**12.76**	**12.40**
+CWS$_{K=4}$	55.92	53	64	**14.45**	**5.24**	**16.49**	**5.20**	**23.12**	**12.99**	**12.92**
+CWS$_{K=8}$	28.46	53	57	**14.33**	**4.94**	**16.06**	**5.08**	**22.77**	**12.70**	**12.65**

Table 3: *Comparison with the state-of-the-art results*

Model	Params (M)	Extra Data	SDR (A) (dB)	SDR (V) (dB)
MMDenseNet [8]	0.33	✓	15.41	3.87
BLSTM [21]	30.03	✓	14.51	3.43
MMDenseLSTM [12]	1.22	✓	16.40	4.94
Spleeter-2stem [25]	19.6	✓	12.88	4.72
MDN	0.27	✗	13.90	2.59
MDN$_{K=8}$	0.27	✗	**15.67**	**4.17**
UNET-5$_{K=8}$	13.3	✗	15.72	4.31
BD-UNET-6$_{K=4}$	53	✓	**16.49**	**5.20**

3.2. Evaluation Metrics

We use *museval* [23] toolkit to compute SDR, SIR, and SAR [24] metrics and perform evaluation. In details, we calculate the metrics for all the segments of the song in the test set with a window size of 1s and hop length of 1s, as was commonly used in SiSEC 2018 [23]. We aggregate both the average SDR, SIR, and SAR by frames as the final score of a song. And pick the median value from each song as the final score of test set. All our experiments are performed on a single GTX 1080Ti GPU. For better and fair comparison, we report some other metrics, e.g., parameter number and training time, as is shown in Table 2. We also use Giga Floating Point Operations (GFLOPs) to weight the computational cost. The floating operation here is measured by a three-second long stereo input.

3.3. Result Comparison

The result is shown in Table 2. We named MMDN and MDN for the abbreviation of MMDenseNet and MDenseNet. UNET-N denotes scale N UNet and the prefix BD means the model is trained with extra internal *aslp* dataset. Value K stands for total subband number in CWS, as was shown in Fig 3.

Generally, the result shows the CWS input can considerably improve the performance of the model. All the models with CWS surpass model without CWS on the average SAR, SDR, and SIR by a large margin. Since the computational cost drops drastically with the increase of K, the model with a higher K value will converge more quickly. This is essential when the dataset is huge. Besides, a higher K will lead to a smaller feature map. This can save a lot of memory during training and evaluation, making the model and training process more flexible and easier to deploy.

As can be seen from Table 2, splitting 4 bands usually has the best average score on all the evaluation metrics. The average performance of MDN, UNET-5, UNET-6 and BD-UNET-6 increase by 5.7%, 10.1%, 10.2%, and 7.8%, using the CWS$_{K=4}$ input. Although CWS$_{K=4}$ outperforms CWS$_{K=8}$ by 1.5%, it takes more time, i.e., 38.5% for model training. Comparing with the model without CWS, CWS$_{K=8}$ increases the performance by 6.8% and costs only 31.8% of the original training time. Also, the experiments on UNET-5/6 and MDN with CWS$_{K=8}$ achieve the best average SDR, which is valid as a global performance measurement [24]. Thus in practice, $CWS_{K=8}$ may be the most effective one because it can yield comparable results in a shorter time. CWS$_{K=2}$ scenario might be the least preferred setting but still it has contributions to the final score.

It's also worth to mention that the MDN$_{K=8}$ surpasses the performance of MMDenseNet in [12] even with fewer parameters, far less training data and shorter training time, as shown in Table 3. The training set only has 84 songs but still able to exceed the performance of the model trained in a larger dataset. Also, our training time might be much shorter. In [8], a single MDenseNet trained on DSD100 [11] dataset, which is comprised of 100 songs, will take 37 hours to train. It's a safe bet that MMDenseNet trained with extra data will cost more than that time. But our model only takes 9.7 hours to train. All the evidence strongly demonstrates the advantage of using CWS as model input. Audio samples are available online: https://haoheliu.github.io/Channel-wise-Subband-Input/.

4. Conclusions

We present an alternative structure of input feature, namely channel-wise subband (CWS) for VAS model in the frequency domain, in order to handle the high computational cost and limitation of the conventional CNNs in high-resolution MSS tasks. It overcomes the limitation of the widely used full-band approach and enables the model to learn weight independently in each subband. Experimental results show that the CWS could improve the separation performance and reduce the computational cost significantly. On the public musdb18hq dataset, the MDenseNet with 8-bands CWS input exceeds original MDenseNet by 1.67 dB on average SDR of the voice and accompaniment.

5. References

[1] E. Cano, D. FitzGerald, A. Liutkus, M. D. Plumbley, and F.-R. Stöter, "Musical source separation: An introduction," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 31–40, 2018.

[2] J. Perez-Lapillo, O. Galkin, and T. Weyde, "Improving singing voice separation with the wave-u-net using minimum hyperspherical energy," *arXiv preprint arXiv:1910.10071*, 2019.

[3] V. S. Kadandale, J. F. Montesinos, G. Haro, and E. Gómez, "Multi-task u-net for music source separation," *arXiv preprint arXiv:2003.10414*, 2020.

[4] Y. Liu, B. Thoshkahna, A. Milani, and T. Kristjansson, "Voice and accompaniment separation in music using self-attention convolutional neural network," *arXiv preprint arXiv:2003.08954*, 2020.

[5] Y. LeCun and Y. Bengio, *Convolutional Networks for Images, Speech, and Time Series*. Cambridge, MA, USA: MIT Press, 1998, p. 255–258.

[6] A. Jansson, E. Humphrey, N. Montecchio, R. Bittner, A. Kumar, and T. Weyde, "Singing voice separation with deep u-net convolutional networks," 2017.

[7] W. Choi, M. Kim, J. Chung, and D. L. S. Jung, "Investigating deep neural transformations for spectrogram-based musical source separation," *arXiv preprint arXiv:1912.02591*, 2019.

[8] N. Takahashi and Y. Mitsufuji, "Multi-scale multi-band densenets for audio source separation," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2017, pp. 21–25.

[9] J. Taghia and M. A. Doostari, "Subband-based single-channel source separation of instantaneous audio mixtures," *World Applied Sciences Journal*, vol. 6, no. 6, pp. 784–792, 2009.

[10] N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Shih, Q. Zheng, N. Yen, C. C. Tung, and H. H. Liu, "The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis," *Proceedings of The Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1971, pp. 903–995, 1998.

[11] A. Liutkus, F. Stoter, Z. Rafii, D. Kitamura, B. Rivet, N. Ito, N. Ono, and J. Fontecave, "The 2016 signal separation evaluation campaign," vol. 10169, pp. 323–332, 2017.

[12] N. Takahashi, N. Goswami, and Y. Mitsufuji, "Mmdenselstm: An efficient combination of convolutional and recurrent neural networks for audio source separation," in *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*. IEEE, 2018, pp. 106–110.

[13] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "Musdb18-hq - an uncompressed version of musdb18," Aug. 2019.

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," pp. 234–241, 2015.

[15] P. Huang, M. Kim, M. Hasegawajohnson, and P. Smaragdis, "Deep learning for monaural speech separation," pp. 1562–1566, 2014.

[16] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," vol. 15, pp. 315–323, 2011.

[17] G. Huang, Z. Liu, L. V. Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," pp. 2261–2269, 2017.

[18] C. Yu, H. Lu, N. Hu, M. Yu, C. Weng, K. Xu, P. Liu, D. Tuo, S. Kang, G. Lei *et al.*, "Durian: Duration informed attention network for multimodal synthesis." *arXiv: Computation and Language*, 2019.

[19] I. Moazzen and P. Agathoklis, "A general approach for filter bank design using optimization," *IET Journal on Signal Processing (Submitted)*, 2014.

[20] P. Chandna, M. Blaauw, J. Bonada, and E. Gomez, "Content based singing voice extraction from a musical mixture," *arXiv: Audio and Speech Processing*, 2020.

[21] S. Uhlich, M. Porcu, F. Giron, M. Enenkl, T. Kemp, N. Takahashi, and Y. Mitsufuji, "Improving music source separation based on deep neural networks through data augmentation and network blending," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 261–265.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv: Learning*, 2014.

[23] F.-R. Stöter, A. Liutkus, and N. Ito, "The 2018 signal separation evaluation campaign," in *International Conference on Latent Variable Analysis and Signal Separation*. Springer, 2018, pp. 293–305.

[24] E. Vincent, M. G. Jafari, S. A. Abdallah, M. D. Plumbley, and M. E. Davies, "Performance measurement in blind audio source separation," *Queen Mary, University of London, Tech Report C4DM-TR-05-01*, 2005.

[25] R. Hennequin, A. Khlif, F. Voituret, and M. Moussalam, "Spleeter: A fast and state-of-the art music source separation tool with pre-trained models," in *Proc. International Society for Music Information Retrieval Conference*, 2019.